Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of queuing threads among ~~a plurality of~~ processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the method comprising the computer implemented steps of:

receiving a first thread to be processed;

identifying the first thread as part of an existing process on a first multi-processor module of the plurality of multi-processor modules; and

performing a search for an idle processor, wherein the search is restricted to the plurality of processors of [[a]] the first multi-processor module associated with the existing process; and

assigning the first thread to either one of the plurality of chip run queues, or one of the plurality of local run queues.

2. (Currently Amended) The method of claim 1, further comprising:

assigning the first thread to [[a]]the chip run queue dedicated to the first multi-processor module.

3. (Original) The method of claim 2, further comprising:

identifying the first multi-processor module as associated with the existing process.

4. (Currently Amended) The method of claim 3, wherein the step of identifying the first multi-processor module further comprises:

maintaining a record of processes having threads executed by a processor of the plurality of processors of the first multi-processor module during a predetermined preceding interval.

5. (Currently Amended) The method of claim 1, further comprising:

identifying one of the plurality of processors of the first multi-processor module as an idle processor; and

assigning the first thread to a local run queue of the plurality of local run queues associated with the idle processor.

6. (Original) The method of claim 1, wherein the step of identifying further comprises:

reading attribute information of the first thread.

7. (Currently Amended) A method of load balancing threads among processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the method comprising the computer implemented steps of:

performing, by an idle processor of the plurality of processors of a first multi-processor module of the plurality of multi-processor modules, a first attempt at a thread steal from a first one of the plurality of local run ~~queue of a processor~~ queues of one of the plurality of processors located on the first multi-processor module for reassignment of a thread to a second one of the plurality of local run queue [[of]] associated with the idle processor; [[and]]

responsive to failure of the first attempt, performing a second attempt at a thread steal from ~~a dedicated queue~~ one of the plurality of chip run queues associated with a second multi-processor module of the plurality of multi-processor modules; and

assigning the first thread to either one of the plurality of chip run queues, or one of the plurality of local run queues.

8. (Currently Amended) The method of claim 7, further comprising:

evaluating a criterion associated with the second multi-processor module; and

responsive to evaluating the criterion, determining if a thread is to be reassigned from the ~~dedicated queue~~ one of the plurality of chip run queues to the local run queue of the idle processor.

9. (Currently Amended) The method of claim 7, further comprising:

reassigning a thread of the ~~dedicated queue~~ one of the plurality of chip run queues to the local run queue of the idle processor.

10. (Currently Amended) The method of claim 7, further comprising:

responsive to failure of the second attempt, performing a third attempt at a thread steal from [[a]] one of the plurality of local run ~~queue~~ queues associated with ~~a processor~~ one of the plurality of the second multi-processor module for reassignment of a thread to the second one of the plurality of local run ~~queue of~~ queues associated with the idle processor.

11. (Currently Amended) A method of load balancing processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the method comprising the computer implemented steps of:

comparing a first thread load of a first chip run queue of the plurality of chip run queues dedicated to a first multi-processor module of the plurality of multi-processor modules with a second thread load of a second chip run queue of a plurality of chip run queues dedicated to a second multi-processor module of the plurality of multi-processor modules; and

reassigning a thread of the first chip run queue to the second chip run queue.

12. (Currently Amended) The method of claim 11, wherein the step of comparing further comprises:

determining a difference between the first thread load of the first chip run queue and the second thread load of the second chip run queue, reassigning the thread responsive to evaluating the difference as greater than a threshold.

13. (Currently Amended) A computer program product in a computer readable medium for queuing threads among processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the computer program product comprising:

first instructions for receiving a first thread to be processed; and

second instructions for assigning the first thread to a first chip run queue dedicated to a first multi-processor module of [[a]] the plurality of multi-processor modules.

14. (Original) The computer program product of claim 13, further comprising:

third instructions for identifying a process associated with the first thread, wherein the second instructions identify threads of the process assigned to the first multi-processor module.

15. (Currently Amended) The computer program product of claim 13, further comprising:

third instructions for comparing a first thread load of the first chip run queue with a second thread load of a second chip run queue dedicated to a second multi-processor module of the plurality of multi-processor modules; and

fourth instructions for reassigning the first thread to the second chip run queue.

16. (Currently Amended) The computer program product of claim 13, further comprising:

third instructions for reassigning the first thread to a ~~second~~ first local run queue dedicated to ~~a processor of~~ one of the plurality of processors for a second multi-processor module of the plurality of multi-processor modules.

17. (Currently Amended) A multiple processor data processing system for executing multi-threaded processes, comprising:

a memory that contains a scheduler as a set of instructions;

a first multi-processor module; and

a second multi-processor module, wherein the scheduler, responsive to execution of the set of instructions, is adapted to receive a thread and assign the thread to a first chip run queue dedicated to the first multi-processor module.

18. (Original) The data processing system of claim 17, wherein the first multi-processor module comprises a plurality of central processing units disposed on a first chip, and the second multi-processor module comprises a plurality of central processing units disposed on a second chip.

19. (Original) The data processing system of claim 17, wherein the first multi-processor module is a simultaneous multi-threading central processing unit, and the second multi-processor module is a simultaneous multi-threading central processing unit.

20. (Currently Amended) The data processing system of claim 17, wherein the scheduler identifies a second thread of a process associated with the first thread, and the second thread is assigned to the first chip run queue.